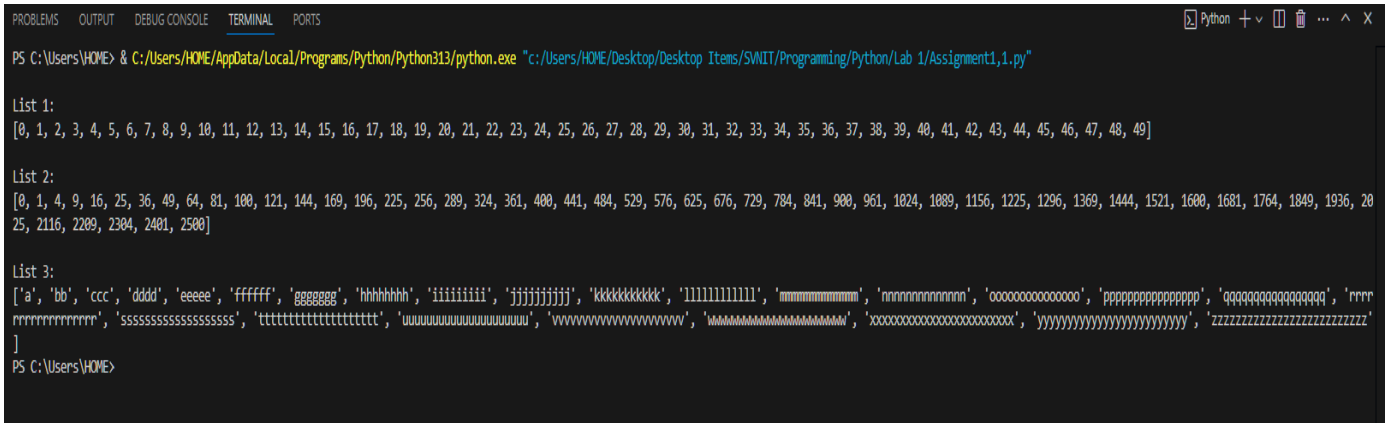
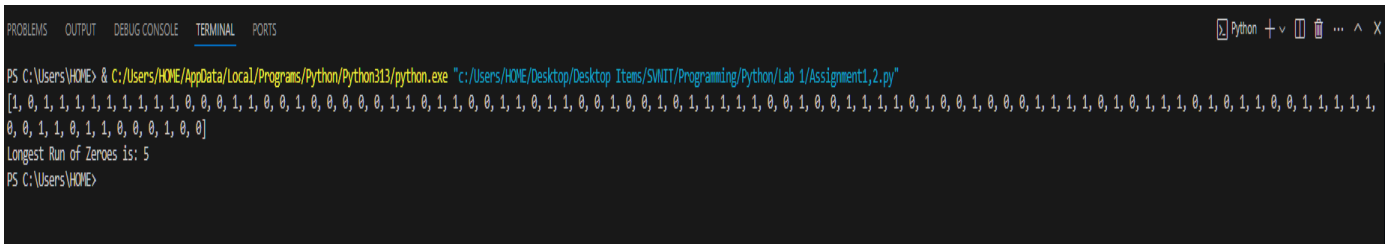
```
list1 = [num for num in range(0,50)]

print("\nList 1:")

print(list1)


list2 = [num**2 for num in range(51)]

print("\nList 2:")

print(list2)


list3 = [str(chr(i))*(i - 96) for i in range(97,123)]

print("\nList 3:")

print(list3)
```

Output:



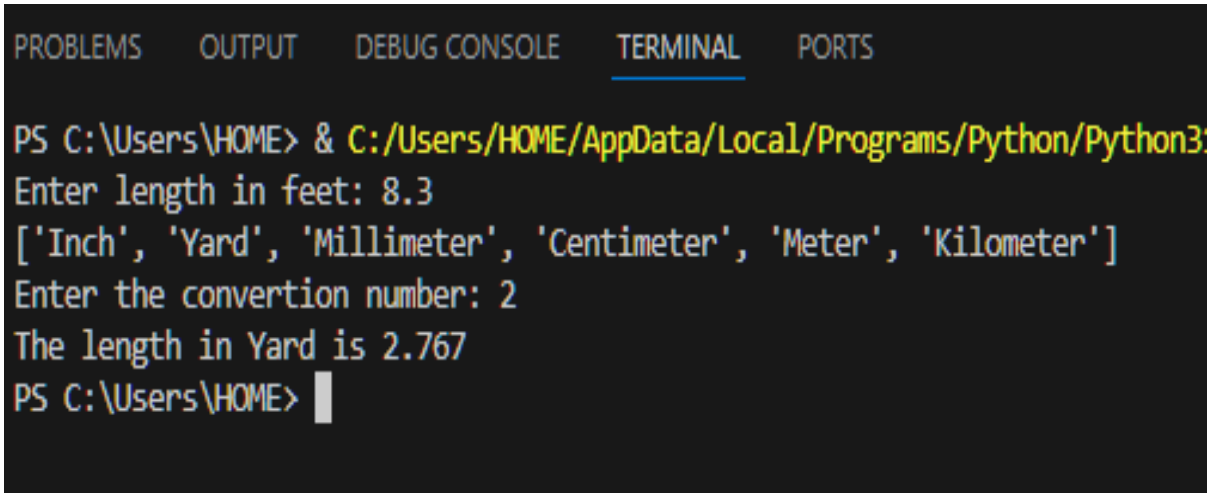
```
import random
mylist = []
count = 0
maxzeroes = 0
for i in range(1,101):
    random_num = random.randint(0,1)
    mylist.append(random_num)
    if (random_num == 0):
        count += 1
    else:
        if (maxzeroes < count):
            maxzeroes = count
        count = 0
print(mylist)
print(f"Longest Run of Zeroes is: {maxzeroes}")
```

Output:



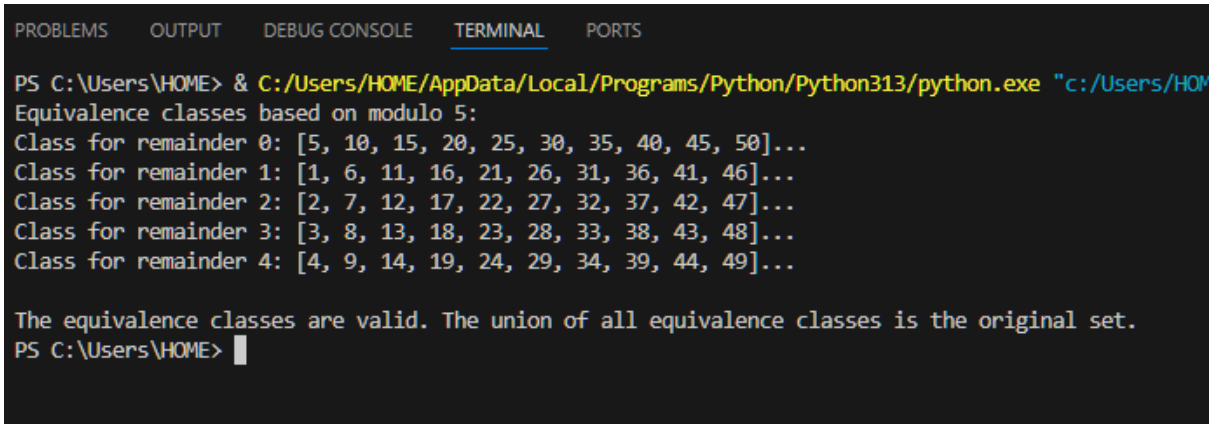
```
inch = 12

yard = 1/3

millimeter = 304.8

centimeter = 30.48

meter = 1000/3281

kilometer = 1/3281


convert = [inch, yard, millimeter, centimeter, meter, kilometer]

convertions =["Inch", "Yard", "Millimeter", "Centimeter", "Meter", "Kilometer"]

feet = float(input("Enter length in feet: "))

print(convertions)

convertion = int(input("Enter the convertion number: "))

print(f"The length in {convertions[convertion-1]} is {(feet * convert[convertion-1]):.3f}")
```

Output:



```
numbers = list(range(1, 10001))

equivalence_classes = {0: [], 1: [], 2: [], 3: [], 4: []}

for num in numbers:
    remainder = num % 5
    equivalence_classes[remainder].append(num)

all_numbers_in_classes = sum(equivalence_classes.values(), [])

is_valid = set(all_numbers_in_classes) == set(numbers)

print("Equivalence classes based on modulo 5:")
for remainder, class_list in equivalence_classes.items():
    print(f"Class for remainder {remainder}: {class_list[:10]}...")

if is_valid:
    print("\nThe equivalence classes are valid. The union of all equivalence classes is the original set.")
else:
    print("\nThe equivalence classes are not valid.")
```

Output:

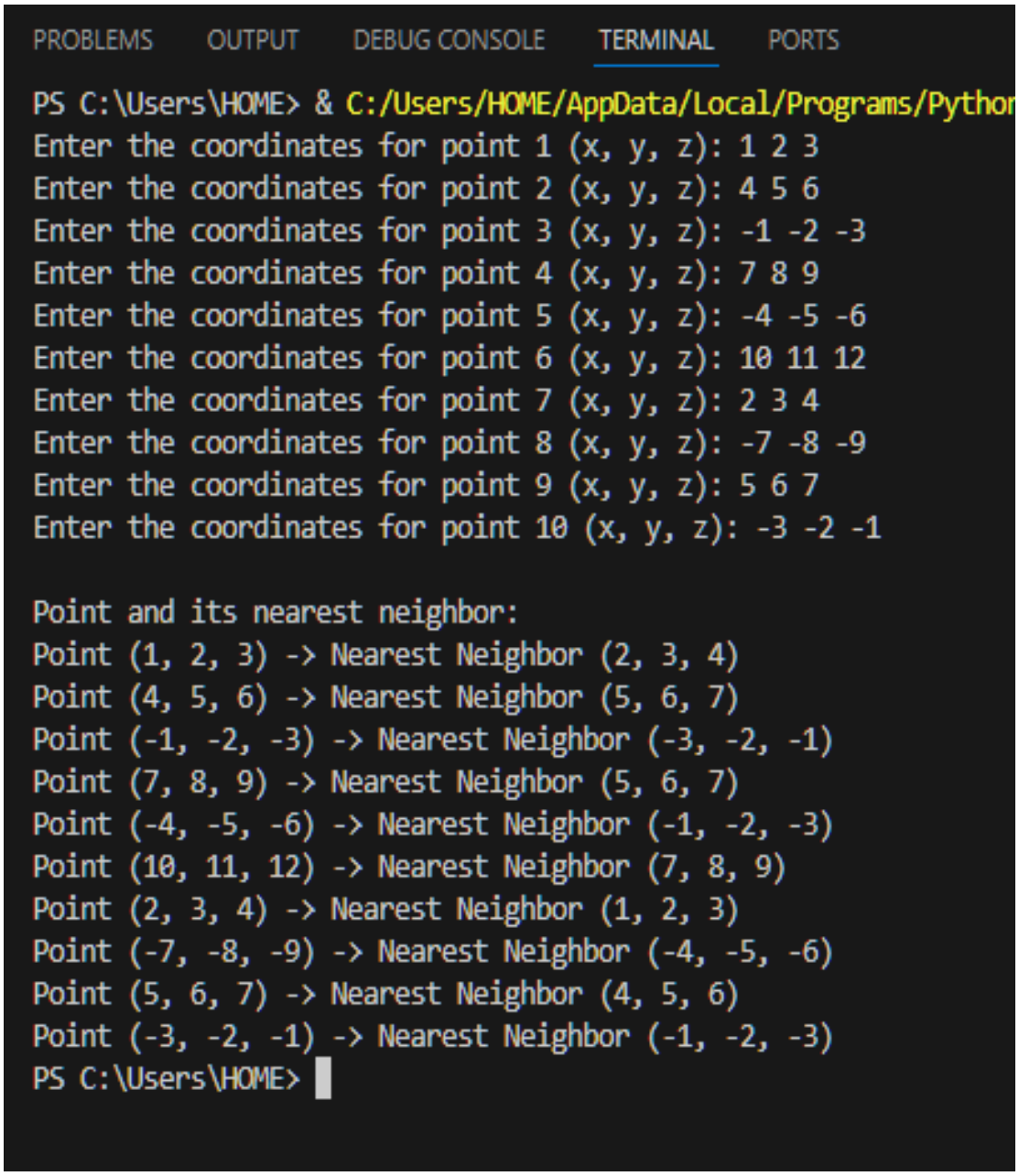


```
names = []
reversed_names = []

for i in range(1,11):
    name = str(input(f"Enter name {i}: "))
    while (len(name) > 15):
        name = str(input(f"Enter a name with less than 15 characters: "))
    names.append(name)

reversed_names = [name[::-1] for name in names]

print("All Names Reversed:")
print(reversed_names)
```

Output:

```
import math

def euclidean_distance(point1, point2):
    return math.sqrt((point2[0] - point1[0])**2 + (point2[1] - point1[1])**2 + (point2[2] - point1[2])**2)

points = []
for i in range(10):
    x, y, z = map(int, input(f"Enter the coordinates for point {i + 1} (x, y, z): ").split())
    points.append((x, y, z))

nearest_neighbors = []

for i, point in enumerate(points):
    min_distance = float('inf')
    nearest_point = None

    for j, other_point in enumerate(points):
        if i != j:
            distance = euclidean_distance(point, other_point)
            if distance < min_distance:
                min_distance = distance
                nearest_point = other_point

    nearest_neighbors.append((point, nearest_point))

print("\nPoint and its nearest neighbor:")
for point, neighbor in nearest_neighbors:
    print(f"Point {point} -> Nearest Neighbor {neighbor}")
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\HOME> & C:/Users/HOME/AppData/Local/Programs/Pythor
Enter the coordinates for point 1 (x, y, z): 1 2 3
Enter the coordinates for point 2 (x, y, z): 4 5 6
Enter the coordinates for point 3 (x, y, z): -1 -2 -3
Enter the coordinates for point 4 (x, y, z): 7 8 9
Enter the coordinates for point 5 (x, y, z): -4 -5 -6
Enter the coordinates for point 6 (x, y, z): 10 11 12
Enter the coordinates for point 7 (x, y, z): 2 3 4
Enter the coordinates for point 8 (x, y, z): -7 -8 -9
Enter the coordinates for point 9 (x, y, z): 5 6 7
Enter the coordinates for point 10 (x, y, z): -3 -2 -1
Point and its nearest neighbor:
Point (1, 2, 3) -> Nearest Neighbor (2, 3, 4)
Point (4, 5, 6) -> Nearest Neighbor (5, 6, 7)
Point (-1, -2, -3) -> Nearest Neighbor (-3, -2, -1)
Point (7, 8, 9) -> Nearest Neighbor (5, 6, 7)
Point (-4, -5, -6) -> Nearest Neighbor (-1, -2, -3)
Point (10, 11, 12) -> Nearest Neighbor (7, 8, 9)
Point (2, 3, 4) -> Nearest Neighbor (1, 2, 3)
Point (-7, -8, -9) -> Nearest Neighbor (-4, -5, -6)
Point (5, 6, 7) -> Nearest Neighbor (4, 5, 6)
Point (-3, -2, -1) -> Nearest Neighbor (-1, -2, -3)
PS C:\Users\HOME>